View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Tsukuba Repository

Efficiently Modeling 3D Scenes from a Single Image

著者	Iizuka Satoshi, Kanamori Yoshihiro, Mitani Jun, Fukui Yukio
journal or publication title	IEEE computer graphics and applications
volume	32
number	6
page range	18-25
year	2012-11
権利	(C) 2012 IEEE Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
URL	http://hdl.handle.net/2241/118109

doi: 10.1109/MCG.2011.85

Efficiently Modeling 3D Scenes from a Single Image

Satoshi Iizuka[1] Yoshihiro Kanamori[1] Jun Mitani[1,2] Yukio Fukui[1]
[1]University of Tsukuba [2]JST ERATO
e-mail: iizuka@npal.cs.tsukuba.ac.jp {kanamori, mitani, fukui}@cs.tsukuba.ac.jp

Abstract— **This paper presents a system for assisting the user to create a 3D model easily and quickly from a single image. Our scene model is composed of a background and foreground objects whose coordinates are calculated based on a "boundary" between "ground" and "wall". Moreover, we introduce a fast method for extracting a foreground object by combining image segmentation and graph cut-based optimization. We show that the proposed system enables efficient modeling of foreground objects, easy creation of their textures, and rapid construction of a 3D scene model that is simple but produces sufficient 3D effects.**

Keywords: image-based modeling/rendering, single view modeling, foreground extraction

1 INTRODUCTION

Recently, approaches for creating a photorealistic scene based on image-based rendering have been of much attention. In these methods, realistic 3D scenes can be generated without modeling accurate geometry thanks to the use of images as textures. By moving the viewpoint in the scene, users can experience a virtual walkthrough in the image. These methods are often used for a three-dimensional representation of a scene and have been adopted for general applications such as Microsoft Photosynth or the street views from Google maps. However, the typical previous methods often demand a dozen numbers of photographs or special devices, or require complicated and time-consuming tasks.

In this paper, we present a system for creating a 3D scene with simple user inputs from a single image. Our scene model is composed of a background model and foreground models made of simple textured planner polygons. In order to calculate the coordinates of the 3D model, the user interactively specifies the boundary with a polygonal line between "ground" and "wall". Based on the boundary, depth is assigned to each model. We also propose a method for fast extraction of foreground objects using image segmentation and graph cut optimization to model the foreground objects. Although these models are simple and do not reconstruct accurate 3D geometry of the input image, the user can obtain convincing 3D effects through a walkthrough animation produced by our system.

Our method requires only a single image as input. Thus, even a single-view image, e.g., an oversea landscape image from the Internet or a scenery painting, suffices to allow the user to walk around in the virtual 3D scene. Of course, our system does not always work well. However, compared to existing methods, our system can handle various types of landscape images interactively and can be a useful tool for constructing a walkthrough animation from an image.

2 RELATED WORK

There are many research studies for constructing a 3D model using image-based rendering. The recent general applications such as the street views from Google maps can construct a wide range of a scene using a large number of photographs overlapping with each other. Similarly, there have been several methods that construct 3D models from multiple photographs or movies with user interactions for specifying object shapes or depth [Debevec et al. 1996; Chen et al. 2011]. These methods, however, need considerable user interaction or are hard to be adopted in case that only a single or a few images are available. Compared to these approaches, making a 3D model from only a single image is more difficult because of the less information about the scene.

Several methods to create a 3D model from a single image have been proposed previously. Hoiem et al. [2005] proposed a fully automatic method based on machine learning and image segmentation. Their method classifies regions in the image into three categories ("ground", "sky" and "vertical") and then creates a 3D model using the regional information. Another method of automatic model construction, *Make3D* [Saxena et al. 2009] uses a Markov Random Field to estimate a correct 3D model structure from a single still image. Although above-mentioned methods are fully automatic, they often fail to reconstruct the scene structure due to errors of depth estimation or image segmentation. Moreover, these methods are difficult to model foreground objects. Unlike these methods, Zhang et al. [2001] described a modeling method of free-form scenes that are constructed of curved surfaces using a sparse set of user-specified constraints on the shape of the scene. In this method, the user should specify many constraints such as surface positions to make a 3D model, and the interface is less intuitive. Oh et al. [2001] proposed an interactive system for modeling a 3D scene by representing the scene as a layered collection of depth images. This system allows users to edit the shape, color, and illumination of the image-based objects, but the whole process imposes significant manual labors (e.g., segmentation of the image into layers, creation of textures using *a clone brushing tool*, and the depth assignment) in editing the image. The *Tour Into the Picture* (TIP) [Horry et al. 1997] is a simple

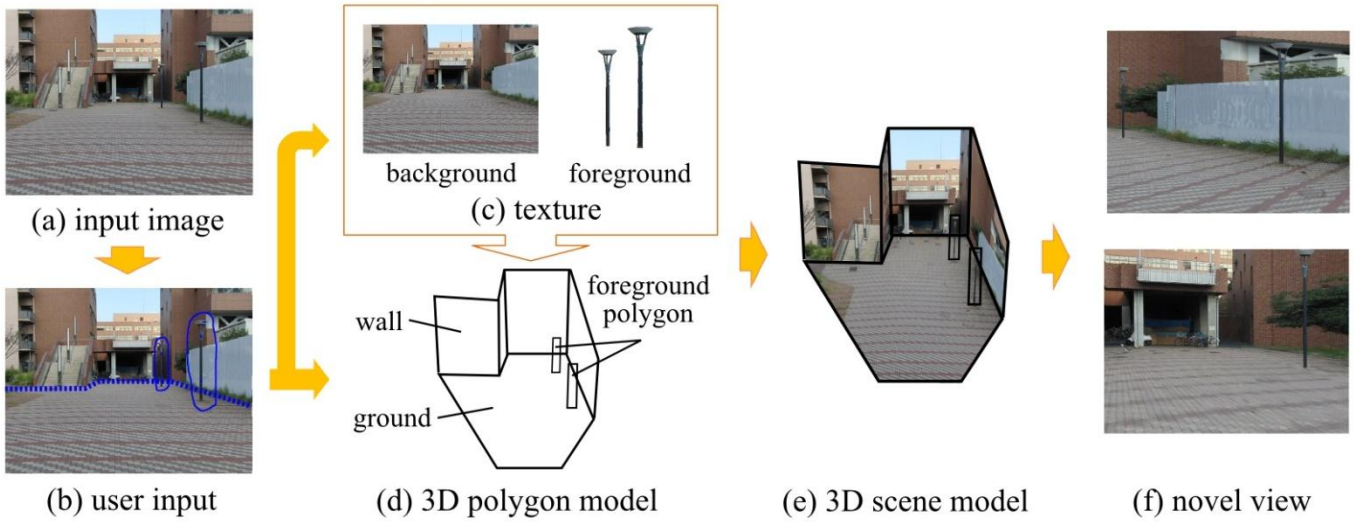


Figure 1: Overview of our 3D scene construction. (a) Given a single image, the user specifies (b) the boundary by a polyline (blue dashed line) and lassoes foreground objects by a brush interface (blue line). Then, (c) the texture images of background and foreground objects and (d) a 3D polygon model is constructed at once. (e) The 3D scene model is generated by mapping the textures onto the corresponding polygons. (f) The user can experience walkthroughs in the scene.

user-guided method that constructs a simple 3D model with five polygons (*floor, right wall, left wall, rear wall,* and *ceiling*) based on a vanishing point. Kang et al. [2001] improved the TIP by using a vanishing line rather than a vanishing point to handle more various types of outdoor images including horizons. The use of horizons, however, impedes applications of this method to urban or indoor scenes where complex structures often appear. Furthermore, preparing textures is the most time-consuming manual process.

In this paper, we propose an efficient single-view modeling system that semi-automatically accomplishes the whole processes for making walkthrough animations, including accurate foreground extraction, successive image completion (*inpainting*) for the background texture as well as calculation of the scene geometry. These processes are performed with quite simple user inputs and do not require complicated 3D operations; the user just has to draw a few strokes on the image, i.e., to draw a polygonal line on the boundary between "ground" and "wall", and to place a lasso around the object.

3 System Overview

Our goal is to obtain 3D walkthrough animations from a single image with simple and intuitive user operations. Particularly, inspired by the pioneering work "*Tour Into the Picture*" (TIP) [Horry et al. 1997], we focus on construction of scene models that are simple but provides convincing 3D effects, instead of modeling complicated geometries accurately. The main contribution of this paper is the overall design of an interactive system that allows construction of such models from simple and intuitive user inputs, without requiring technical skills on 3D modeling and image editing.

Given a single scenery image including the flat ground, our system constructs a simple 3D scene consisting of a background model and foreground models made of textured polygons. The foreground models are vertically placed on the ground region of the background model. To make a scene model, the user employs only two simple tasks;

1) drawing of a boundary with a polygonal line, and

2) lassoing the foreground object coarsely for extracting them.

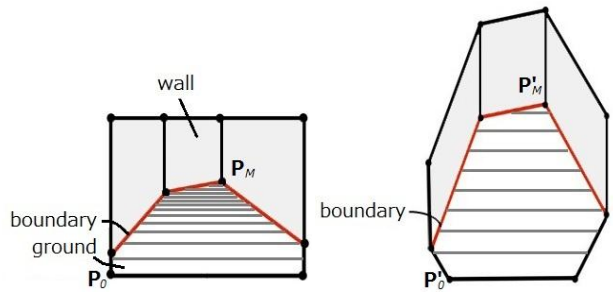
These tasks are fully performed on the 2D image and do not require any complex 3D operations. Figure 1 shows the overview of our system.

3.1 Definition of the boundary line

The scene model is determined by the boundary that is interactively specified by the user. In this paper, we define the boundary as a polygonal line that separates the image into the ground plane and wall plane. For example, in Figure 1(b), the boundary separates the scene into the ground and buildings. The buildings and sky in the image are represented as wall planes in the model. In the case where the boundary becomes a curve as illustrated in Figure 8(e), the boundary can be approximately specified by adding more vertices to the polygonal line. Based on the boundary, our system determines the 3D structure of the background model and the positions of foreground models (see Figure 1(e)). Moreover, the boundary is also used as a constraint to improve the completion of extracted foreground regions for making a background texture. The more details are described in Section 4.1.

3.2 Extracting and modeling foreground objects

A foreground object mentioned in this paper is an object that is placed on the ground region, and is modeled as a single planar polygon whose texture is extracted from the corresponding region in the image. In general, accurate



(a) 2D image (b) 3D model

Figure 2: (a) A 2D image and (b) the corresponding 3D model.

extraction of foreground objects is a time-consuming process. Therefore, we propose a fast method to extract foreground regions easily based on image segmentation and graph cut-based optimization. To extract a foreground object, the user roughly lassoes a foreground region and then the system optimizes the region. In case that the extracted region is inaccurate at the first attempt, the user marks the miss-labeled parts with a brush tool to label them as foreground or background. Then, the system refines the extraction according to this additional user input. As a result of these steps, the user can obtain a desired labeling of the foreground object.

4 SCENE MODELING

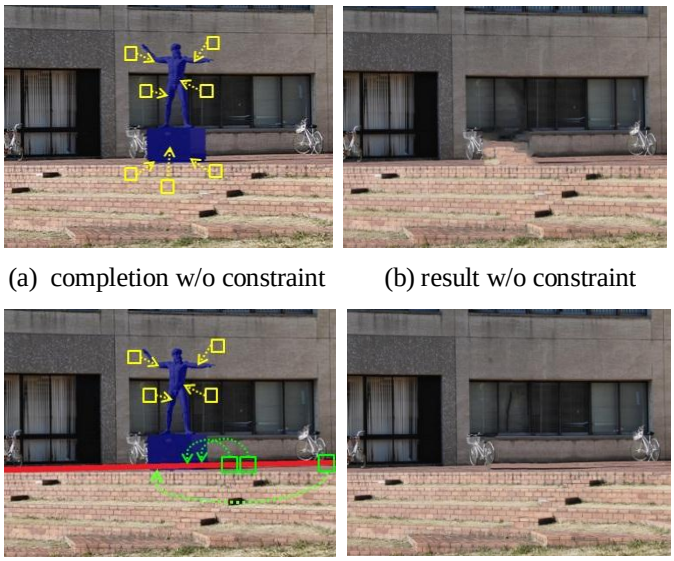
In this section, we introduce the basic algorithms of our system. Section 4.1 describes the algorithm to create the background model, and the way to generate the background texture. Section 4.2 presents methods for fast foreground extraction and model construction with a billboard transform and a ground constraint.

4.1 Background model

The input image is separated into a ground polygon and multiple vertical polygons based on the boundary specified by the user (see Figure 2). The background model is constructed by assigning an appropriate depth value to each vertex of the ground polygon and vertical polygons. The polygon model is then textured with an image synthesized from the input image by extracting the foreground objects and by completing holes caused by extraction.

In our system, we assume that the camera is located at the origin, the view direction is towards $+z$, and the focal length is f that is known. Also, the ground plane intersects the bottom of the image. The model is computed by transforming the coordinates of vertex i $\mathbf{P}_i(x_i, y_i)$ into homogeneous coordinates $\mathbf{P'}_i: (x'_i, y'_i, f, w_i)$. Let the screen coordinate of the bottom-left corner of the image be $\mathbf{P}_0(x_0, y_0)$, and the vertex with the maximum y value among the boundary vertices be $\mathbf{P}_M\ (x_M, y_M)$, respectively. Then, their corresponding homogeneous coordinates are represented as follows:

$$\mathbf{P'}_0: (x'_0, y'_0, f, 1) \qquad \mathbf{P'}_M: (x'_M, y'_M, f, w_{min})$$



(a) completion w/o constraint (b) result w/o constraint

(c) completion w/ constraint (d) result w/ constraint

Figure 3: Synthesis of the background texture by compositing the region behind the object from similar patches (yellow). (a) The statue is labeled as foreground (blue) and (c) the boundary (red) between the ground and the building is specified. The result of completion (d) using similar patches with the boundary constraint (green) is much more natural than (b) that without the constraint.

where w_{min} is zero ideally since we assume that $\mathbf{P'}_M$ is an ideal point. However, an ideal point cannot be displayed, thus we set w_{min} to a fractional value (we use $w_{min} = 0.1$). Based on these two coordinates, each vertex of the ground polygon is computed:

$$\mathbf{P'}_i: (x'_i, y'_i, f, w_i)$$

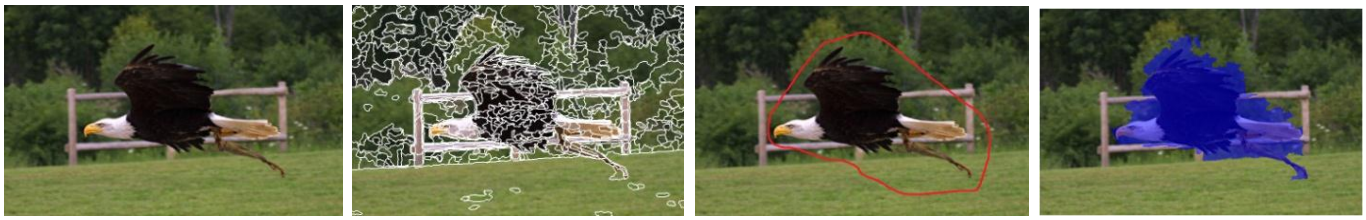
where

$$w_i = \frac{y_i - y_0}{y_M - y_0} w_{min} + 1 - \frac{y_i - y_0}{y_M - y_0} \qquad (1)$$

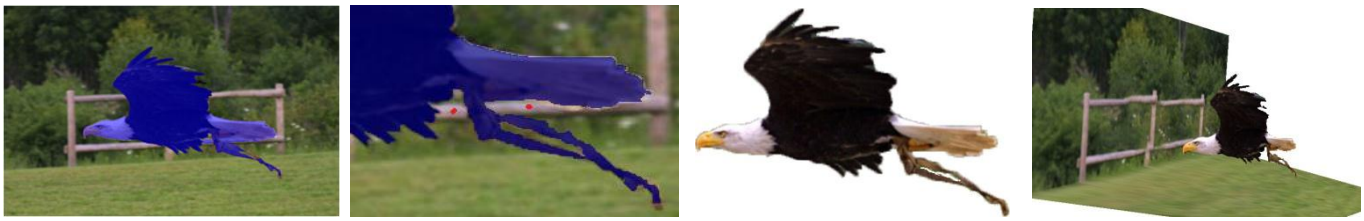
The rest of vertices are calculated based on the constraint that wall polygons are perpendicular to the ground.

Synthesizing the region behind foreground objects

In addition to the calculation of vertex coordinates, we require the texture image for the background model. The background texture image is synthesized from the input image by extracting foreground objects (see Section 4.2) and completing the holes caused by the extraction. For the image completion or *inpainting*, we employ a very fast patch-based method called "PatchMatch" [Barnes et al. 2009] to find appropriate patches from the input image and fill the holes using them. Our system instantly inpaints the hole every time a foreground

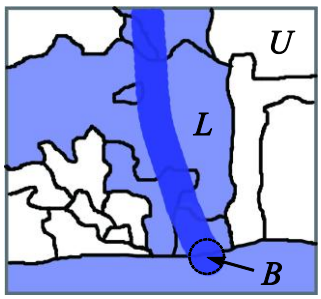


(a) input image (b) segmentation (c) user input (d) region-based selection



(e) graph cut optimization (f) modification (g) extracted foreground (h) novel view

Figure 4: Extraction of a foreground object. (a) The input image is (b) segmented in preprocessing by a mean shift-based approach [Comanniciu and Meer 2002]. (c) The user lassoes a foreground object roughly. (d) The segmented regions outside and across the lasso stroke are labeled as background, and the rest becomes the initial foreground region. After (e) the graph cut-based optimization is performed, (f) further user editing is applied to correct the miss-labeled pixels. Finally, (g) the extracted foreground image is texture-mapped onto the quadrangular polygon to construct the foreground model. (h) The 3D scene is generated by combining the background and foreground object.



B – user brush

L – labeled superpixels

U – uncertain superpixels

Figure 5: Labeling of superpixels using the user stroke.

object is extracted, and thus the background texture is available when the extraction is done.

However, naïve inpainting causes artifacts in the inpainted region when inadequate patches are selected and used (see Figure 3(b)). To avoid this, we constrain the search space of similar patches using the boundary line as a guide as follows. First, the hole region along the boundary is completed with the patches sampled from the non-hole regions along the boundary. Then, the upper part of the hole is completed with patches sampled from the upper region of the boundary, and similarly for the lower part. Consequently, the boundary plays two roles in our system; a reference for the calculation of 3D models and a guide for the synthesis of the background texture.

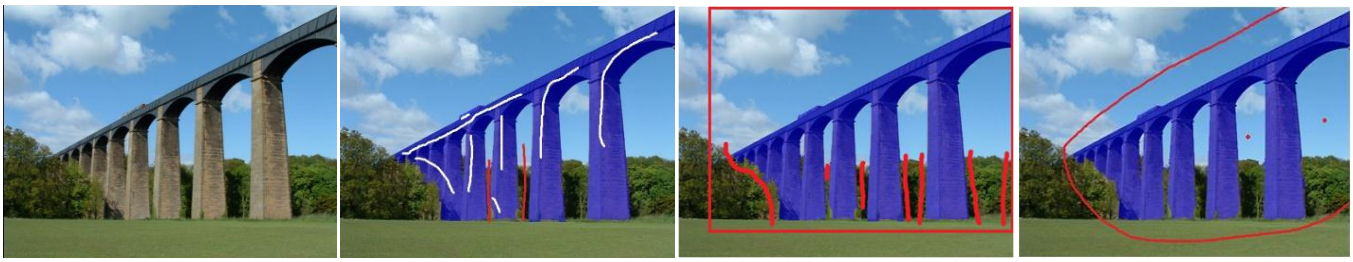
As a metric of patch similarity, we calculate the sum of squared distance for patches of size 7×7 pixels in RGB color space, similarly to [Barnes et al. 2009].

4.2 Foreground model

Extraction of foreground objects is one of the most time-consuming tasks in image editing. Previous research has introduced painting-based approaches that let users directly paint the regions using a brush or boundary-based approaches that trace the region boundary. However, these methods sometimes require accurate and detailed user operations according to the intended foreground shape. To eliminate such labor-intensive operations, we propose a scribble-based selection approach whose interface is just like a lasso tool. Extraction is performed by roughly specifying a possible foreground region based on nearly-uniform regions called superpixels, followed by fine optimization of the region based on graph cut.

As preprocessing, the input image is segmented into superpixels when loaded using mean shift [Comaniciu and Meer 2002]. This method handles each pixel as a 5D vector consisting of the pixel position (*x*, *y*) and color (*L*, *u*, *v*), and applies mean-shift segmentation, yielding high accuracy for clustering similar pixels in various images. We use the segmentation with the parameters $(h_s, h_r, M) = (7, 4, 100)$ in all results in our paper, where h_s is the kernel bandwidth of the spatial domain, h_r is the kernel bandwidth of the range domain of the feature vector, and M is the minimum pixel counts of superpexels. The detailed description of the parameters is shown in [Comaniciu and Meer 2002]. For example, in Figure 4, the input image is segmented into 568 superpexels.

In the editing session, the user loosely encloses a possible foreground region using a lasso tool (see Figure 4(c)). According to the lasso stroke, the labeling of pixels is determined for each superpixel as shown in Figure 5; the superpixels outside and across the stroke are labeled as



(a) input image (b) Photoshop Quick Selection (c) [Rother et al. 2004] (d) our method

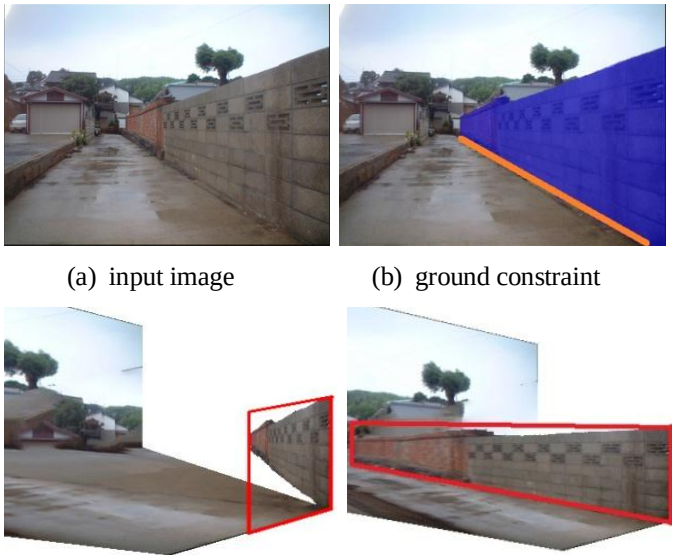
Figure 6: Comparison with other methods. The user labels pixels as "background" with red strokes and "foreground" with white strokes, and obtains the resultant extracted objects in blue. Compared to (b) Photoshop Quick Selection and (c) GrabCut [Rother et al. 2004], (d) our method allows quicker foreground extraction with more rough and fewer sketches.

background (see Figure 4(d)). The inside of the stroke is labeled as unknown, and then optimized by graph cut. Note that we use a Gaussian Mixture Models exactly as GrabCut [Rother et al. 2004]. Whilst the original GrabCut approach often requires several iterations for graph cut and thus is not well suited for an interactive use, our method requires almost no iterations to achieve a sufficient labeling of the foreground region thanks to the pre-segmentation.

In preparation for failure cases at the initial extraction, our system also supplies users with manual correction tools. In our system, the user can correct the miss-labeled regions using rough brush strokes, as shown in Figure 4(f). Then, the labeling of pixels is updated in unit of superpixels computed in preprocessing. The graph cut optimization is performed once after that. These steps are iterated until the foreground region is appropriately extracted. This extraction method can label the foreground region accurately without precise and detailed user operations.

Figure 6 compares our method with *Adobe Photoshop CS5 Quick Selection*, which is a powerful image editing tool the most similar to ours, and our implementation of GrabCut without "border matting". The input image size is 800×600 pixels. In our system, pre-segmentation was done when the image was loaded and took only 0.71 seconds. The time required for our optimization for a single user stroke was 0.41 seconds. The total time for extracting the foreground regions including the user operation time is 107 seconds in Photoshop Quick Selection, 115 seconds in GrabCut, and 18 seconds in our method. Note that the accuracy of our method depends on the graph cut optimization, and thus it does not improve the quality of results than previous graph cut-based approaches like GrabCut. However, compared to the other methods, our system achieves more rough strokes and less user interactions by combining pre-segmentation and graph cut-based optimization.

Care must be taken for the failure cases of image segmentation in preprocessing; foreground and background pixels can be mixed in a single segmented region, which breaks the border between the background and foreground regions. This situation can be avoided by taking advantage of other selection tools provided in our system. The user can apply graph cut without image segmentation or directly paint the regions for labeling. These options make our system applicable to any foreground objects.



(a) input image (b) ground constraint

(c) result without constraint (d) result with constraint

Figure 7: An example of the ground constraint. (b) The wall is labeled as the foreground (blue) and the ground constraint is specified by a line (orange). (c) Without the constraint, the same depth is assigned to the vertices of the foreground polygon. (d) The constraint line modifies the depth to fit the accurate geometry.

The foreground object extracted in this way is modeled as a quadrangular polygon textured with the corresponding foreground image, and is located vertically on the ground. The 3D coordinates of the foreground model are computed based on the pixel coordinate with the minimum y value in the foreground region using Equation (1).

Billboard transformation. Because a foreground object is modeled as a planner polygon, it loses the sense of reality if viewed from the side. To avoid this, our system employs a billboard transformation technique. This method rotates the target polygon so that it faces the viewpoint, and it is applied particularly to a foreground object such as a tree or column-shaped object that commonly appear in landscape images.

(a) input image (b) automatic photo pop-up [Hoiem et al. 2005] (c) spidery mesh [Horry et al. 1996] (d) vanishing line [Kang et al. 2001] (e) boundary line

Figure 8: Typical results from our system and existing methods for creating walkthrough animations. The yellow lines show the user input and the red lines represent the remaining wireframe of each model.

Ground constraint. If a foreground object faces an oblique direction against the camera, the object gets away from the ground because the same depth is assigned to each vertex by default (see Figure 7). Our system provides users with a tool to adjust the depth by specifying the border between the foreground and ground using a polygonal line. This function can handle foreground objects composed of multiple polygons.

4.3 Discussion

Here we discuss the design principle for our system while mentioning the differences between our method and previous methods. Existing automatic methods such as [Hoiem et al. 2005] and [Saxena et al. 2009] often fail to handle foreground objects or separation between the ground and vertical regions. Such tasks are essentially recognition of objects in images according to their semantic characteristics in the real world, which is still quite hard for computers. On the other hand, in the case of human, even a child can easily recognize the boundaries between the ground and walls or between foreground and background. The key idea of our system is to leverage such human recognition capability for model construction via intuitive operations (i.e., drawing of a boundary line and lassoes), and to let the system perform precise operations that human is not good at and might require much labors and technical skills (e.g., accurate foreground extraction and texture synthesis), in order to exploit the advantages of human and computer by combining them.

Recall that the goal of our system is that even casual users can accomplish the whole processes of making 3D walkthrough animations from a single image. The work of Oh et al. [2001] seems excellent as an accurate modeling system for various scenes because it even enables to edit precise geometries of objects and lighting effects in the scene. However, the versatility of their system is at the cost of laborsome manual operations for, e.g., separating the image into layers, assigning depth to each layer and inpainting the holes behind foreground objects. Such versatility and heavy tasks are not appropriate for easy creation of plausible walkthrough animations, which we seek.

The *Tour Into the Picture* [Horry et al.1997; Kang et al. 2001], which is the most similar to our system and shares the same goal, concentrates the construction of 3D polygonal models and leaves the remaining tasks untouched; the user must extract foreground objects and inpaint holes using photo-retouching software, which ends up laborsome manual operations requiring technical skills. Additionally, even for the scenes without foreground objects, the modeling interfaces such as the spidery mesh or the vanishing line cannot handle winding or curved boundaries, as shown in Figure 8(c)(d).

To solve the flaws mentioned above, we propose

1. the boundary interface for more accurate modeling of the background model (see Figure 8(e)), and
2. on-the-fly extraction and inpainting for foreground objects with lassoing.

Both of them are accomplished with simple and intuitive operations, and as a whole our system allows easy creation of 3D walkthrough animations.

5 Results

We implemented our prototype system with C++, OpenGL and GLUT, and ran the program on a PC with Intel Core i7 620M 2.67GHz CPU and 4.00GB RAM. The sizes of input images are all in the range of 0.5 to 1 megapixels.

In Figure 1, two streetlights are specified as foreground and the boundary is specified by a polygonal line with five vertices. As viewed from the side, the streetlights seem tactile due to the billboard transformation. A standard tree or a cylindrical object like the streetlight can be represented by the billboard

transformation. In our scene model, each region such as the ground or wall is modeled as a planner polygon; the staircase and the rear building seem unnatural because they are represented as the same plane.

Figure 8 compares the background modeling of ours and the existing methods. The figure shows that our boundary interface can more accurately separate the ground and walls to model more natural shapes than the automatic pop-up [Hoiem et al. 2005], the spidery mesh interface [Horry et al. 1999] and the vanishing line interface [Kang et al. 2001]. Compared to automatic methods [Hoiem et al. 2005], we emphasize that our human-assisted system offers more flexibility to produce higher quality as discussed in Section 4.3.

Figure 9 shows several types of input images with foreground objects and the walkthroughs in the 3D scenes generated by our system.

Note that the creation time using our system is within 3 minutes (including manual operations) in all examples illustrated in this paper. The most time-consuming part in the process is the extraction of the foreground objects. For example, if foreground extraction is not necessary (the bottom of Figure 8), the creation time is only around 14 seconds, but the time for the scene with foreground objects (the top row of Figure 9) is 127 seconds. The time increases according to the number of foreground objects. For this reason, the fast extraction method in our system plays a very important role in the creation of the models.

6 Conclusion and Futuer Work

In this paper, we have proposed an efficient modeling system for constructing a 3D scene model from a single image. The scene model consists of a background model and foreground models, and the 3D coordinates are computed based on the boundary between "ground" and "wall", which is interactively specified by users. By the modeling scheme using a boundary, we can make 3D scene models from various types of landscape images. Whilst our model does not reconstruct an exact complex geometry in the input image, a realistic walkthrough animation can be generated by navigating the scene. We have also introduced a method for extracting a foreground object easily to model them. This method optimizes the foreground regions roughly specified by the user based on the combination of image segmentation and graph cut-based approach. The extracted foreground region is inpainted automatically to make a background texture. Our method enables users to model a 3D scene from a single image more easily and quickly than previous methods.

Our foreground model is basically represented as a simple planner polygon. In future work, we would like to construct the detailed foreground model using an intuitive sketch-based modeling such as [Chen et al. 2008] for improving the sense of reality. We also plan to improve the quality of the scene texture that is elongated due to the perspective projection, using super-resolution approaches. We believe that our system enriches the user understanding of photos by navigating into the scene.

Acknowledgment

We would like to thank the following Flickr users for Creative Commons imagery: emilio Labrador, ell brown and Jorge BRAZIL.

References

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Trans. Graph. 28, 3, 24:1–11.

CHEN, J., PARIS, S., WANG, J., MATUSIK, W., COHEN, M., AND DURAND, F. 2011. The Video Mesh: A Data Structure for Image-based Three-dimensional Video Editing. In Proceedings of ICCP.

CHEN, X., KANG, S. B., XU, Y.-Q., DORSEY, J., AND SHUM, H. Y. 2008. Sketching reality: Realistic interpretation of architectural designs. ACM Transactions on Graphics 27, 2, 11.

COMANICIU, D. AND MEER, P. 2002. A robust approach toward feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence, 24, 5, 603–619.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. In Proceedings of ACM SIGGRAPH, 11–20.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. ACM Trans. Graph. 24, 3, 577–584.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: using a spidery mesh interface to make animation from a single image. In Proceedings of ACM SIGGRAPH 97, 225–232.

KANG, H., PYO, S., ANJYO, K., AND SHIN, S. 2001. Tour into the picture using a vanishing line and its extension to panoramic images. In Proceedings of Eurographics, 132–141.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In Proceedings of ACM SIGGRAPH, ACM Press, 433–442.

ROTHER, C., BLAKE, A., AND KOLMOGOROV, V. 2004. Grabcut - interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 24, 3, 309–314.

SAXENA, A., SUN, M., AND NG, A.Y. 2009. Make3D: Learning 3D Scene Structure from a Single Still Image. IEEE Trans. Pattern Analysis and Machine Intelligence, 31, 5, 824–840.

ZHANG, L., DUGAS-PHOCION, G., SAMSON, J., AND SEITZ, S. 2001. Single view modeling of free-form scenes. In Proceedings of CVPR, 990–997.

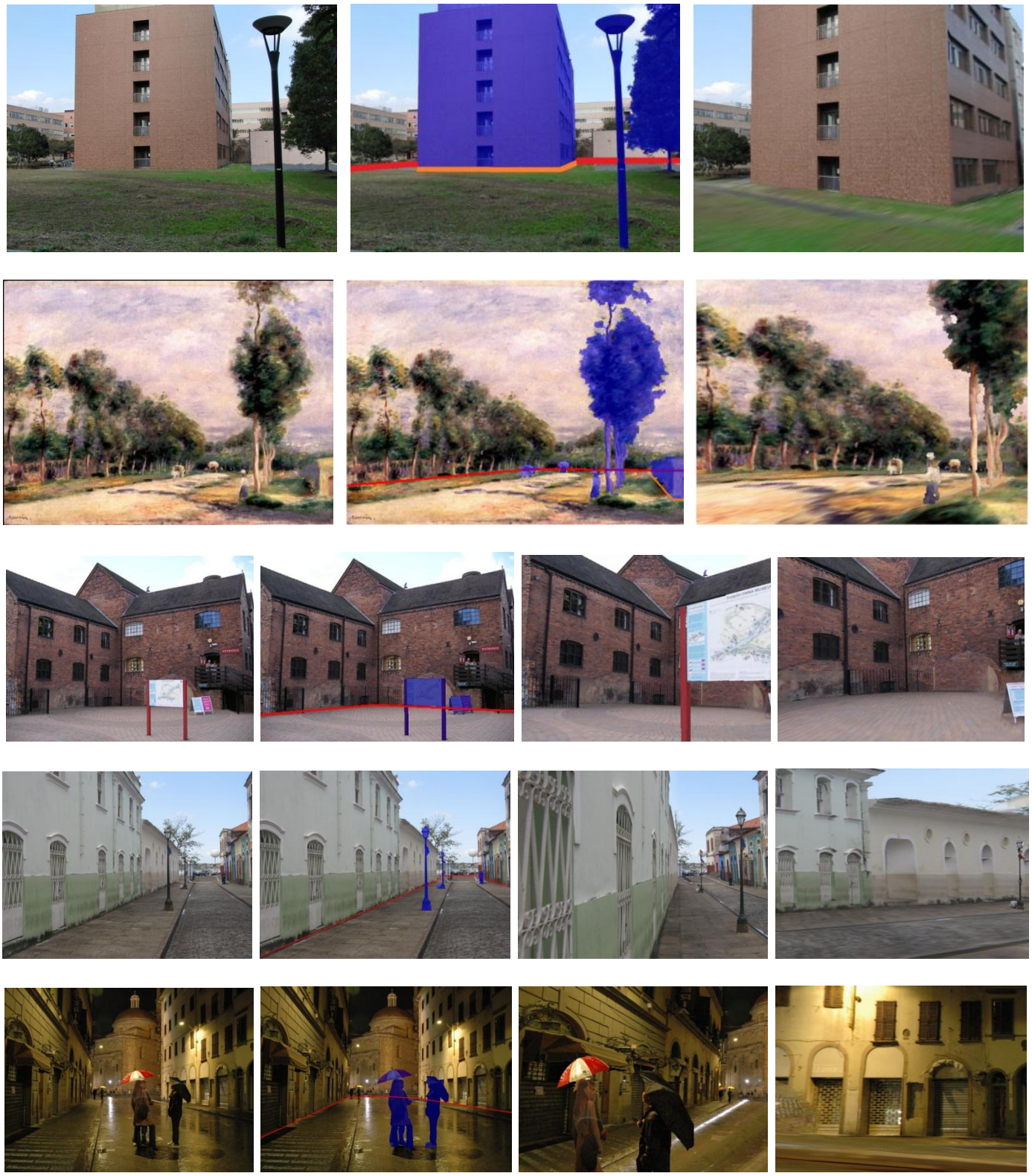


Figure 9: Given input images (the left-most column) and user inputs (the second column from the left), the walkthrough images (the third and forth columns from the left) are generated using our system.

Satoshi Iizuka received his B.S degree in computer science from the University of Tsukuba in 2010, Japan. Since April 2010, he has been a master's degree student in the Department of Computer Science at University of Tsukuba. His research interests center in computer graphics including interactive simulations, image-based modeling and image editing.

mailing adress : University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-8573, Japan
phone/fax : 029-853-5388
Email : iizuka@npal.cs.tsukuba.ac.jp

Yoshihiro Kanamori received his B.S., M.S., and Ph.D. degrees in computer science from the University of Tokyo, Japan, in 2003, 2005, and 2009, respectively. Since April 2009, he has been an assistant professor in the Department of Computer Science at University of Tsukuba. His research interests center in computer graphics including real-time rendering, point-based graphics, and interactive simulations.

mailing adress : University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-8573, Japan
phone/fax : 029-853-5388
Email : kanamori@cs.tsukuba.ac.jp

Jun Mitani In 2004 Mitani received Ph.D. in Engineering from the University of Tokyo for his research on designing 3D papercraft models with computer. In 2006 he started working at University of Tsukuba. His present post is associate professor at Graduate School of Systems and Information Engineering in University of Tsukuba. He is studying about designing three-dimensional curved origami with computer as part of research activities.

mailing adress : University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-8573, Japan
phone/fax : 029-853-5388
Email : mitani@cs.tsukuba.ac.jp

Yukio Fukui received Ph.D. in Engineering from the University of Tokyo in 1993. He had been working at National Institute of Bioscience and Human-Technology, M.I.T.I. and Institute of Information Sciences and Electronics at University of Tsukuba since 1993 and 1998 respectively. He has been a professor in the Department of Computer Science at University of Tsukuba since 2004.

mailing adress : University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-8573, Japan
phone/fax : 029-853-5388
Email : fukui@cs.tsukuba.ac.jp